Xpert.press

Mark Pilgrim
Florian Wollenschein

Python 3
Intensivkurs

Springer

Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Mark Pilgrim

Python 3 – Intensivkurs

Projekte erfolgreich realisieren

Übersetzt aus dem Amerikanischen von
Florian Wollenschein

Springer

Mark Pilgrim
101 Forestcrest Court
Apex NC 27502
USA
mark@diveintomark.org

Übersetzer
Florian Wollenschein
Wertheimer Straße 21
97828 Marktheidenfeld
Deutschland
fw@florianwollenschein.de

Übersetzung aus dem Amerikanischen mit freundlicher Genehmigung des Autors. Titel der amerikanischen Originalausgabe: Dive into Python 3, Apress 2009.

ISBN 978-3-642-04376-5 e-ISBN 978-3-642-04377-2
DOI 10.1007/978-3-642-04377-2
Springer Heidelberg Dordrecht London New York

Xpert.press ISSN 1439-5428

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandentwurf: KünkelLopka GmbH, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

Vorwort

Geschichte einer Übersetzung

Es begann im April 2009. Ich hatte gerade nichts Wichtiges zu tun, da kam mir die Idee, *Dive Into Python* (das englische Original der älteren Version dieses Buches) ins Deutsche zu übersetzen. Erfahrung im Übersetzen und in der Programmierung hatte ich über die letzten Jahre gesammelt. Probleme sollte es also nicht geben. Doch: vielleicht eins. Ich wusste nicht, ob *DIP,* wie ich es mittlerweile nenne, bereits ins Deutsche übersetzt war (immerhin hatte es schon über fünf Jahre auf dem Buckel). Ein Besuch der Homepage brachte aber schnell Klarheit und es stellte sich heraus, dass es bisher keine deutsche Übersetzung gab. Sodann beschloss ich frohen Mutes die Idee in die Tat umzusetzen und schrieb Mark Pilgrim eine E-Mail mit einem Hinweis darauf. Einen Tag (es können auch zwei gewesen sein) später antwortete er auch schon mit der Mitteilung, dass bereits *Dive Into Python 3* in Arbeit sei und ich doch vielleicht lieber das übersetzen solle, da *DIP* veraltet sei und in nächster Zeit auch nicht aktualisiert werde. Gut, dachte ich mir, dann eben *DIP 3*! Je aktueller, desto besser.

Ich begann sofort mit der Übersetzung.

Am 18. Mai 2009 stöberte ich ein wenig im Internet und stieß eher zufällig auf das Impressum der amerikanischen Ausgabe von *DIP*, die bei *Apress* erschienen war. *Springer-Verlag, Heidelberg, Germany* fiel mir sofort ins Auge. Offensichtlich gehörte Apress also zum deutschen Springer-Verlag. Mir kam ein Gedanke, der sich im Nachhinein als nicht so abwegig herausstellte, wie ich anfangs vermutet hatte. Eine E-Mail an Springer musste her. Ich surfte also zu *springer.com* und klickte mich zum Kontaktformular. Ich fragte, ob sie Interesse daran hätten, *Dive Into Python 3* auf Deutsch herauszubringen.

Die Chance überhaupt eine – nicht automatisch generierte – Antwort zu erhalten, schätzte ich damals auf etwa 25 Prozent. Warum gerade 25 Prozent? Sagen wir vielleicht lieber, es erschien mir recht unwahrscheinlich. Springer ist immerhin der zweitgrößte Wissenschaftsverlag der Welt. Warum sollte gerade dieser riesige Verlag sich für die Idee eines jungen Übersetzers erwärmen? Ich sollte eines Besseren belehrt werden. Dorothea Glaunsinger, die Assistentin des für den Informatik-Bereich zuständigen Lektors Hermann Engesser, antwortete bereits nach kurzer Zeit

und teilte mir mit, dass es in der Tat eine sehr interessante Idee sei. Wenn möglich sollte ich ein Probekapitel senden, damit sie sich ein Urteil über meine Übersetzung bilden könnten. Hermann Engesser melde sich dann bei mir.

Ich schickte meine zwei, bereits fertigen, Kapitel per E-Mail an Frau Glaunsinger und war nun aufgeregter denn je. Werden sie die Übersetzung positiv aufnehmen?

(Nun, Sie wissen bereits, wie die Geschichte ausgegangen ist, da Sie dieses Buch nun in Händen halten, doch *ich* wusste es damals nicht.)

Nachdem Frau Glaunsinger mir bereits kurze Zeit später geschrieben hatte, dass die Probekapitel sehr gut seien, teilte mir auch Herr Engesser später sein Urteil mit. *Sehr gut*, meinte auch er. Hätte ich Champagner gehabt – würde ich überhaupt Champagner trinken – hätte ich mir nun eine Flasche aufgemacht. Sie müssen wissen, dass dies mein erster Versuch war, bei einem *richtigen* Verlag zu veröffentlichen. Wenn man dann ein so positives Feedback erhält, ist das wie ein Sechser im Lotto … oder … nein, sagen wir lieber wie ein Fünfer mit Zusatzzahl. Der Sechser kommt erst noch.

Nach weiteren E-Mails kam das erste Telefonat mit Hermann Engesser, meinem Lektor in Spe (und das meine ich in der wörtlichen Übersetzung „in der Hoffnung"). Es wurden hauptsächlich rechtliche Fragen besprochen und ich sollte Mark Pilgrim mitteilen, dass er doch bitte seine Erlaubnis zur Veröffentlichung an Springer sendet. Abschließend fragte ich, wie hoch er die Wahrscheinlichkeit einer Veröffentlichung einschätze. 80:20. Wow! Ich versuchte cool zu bleiben, was mir aber – auch bei unserem zweiten Telefonat – nicht sehr gut gelang. Doch auch Hermann Engesser klang immer so begeistert, dass er mich damit ansteckte.

Nachdem auch die Marketingleute das Projekt abgesegnet hatten, rief mich Herr Engesser am 20. Juli, also fast genau zwei Monate nach meiner ersten Kontaktaufnahme, erneut an. Er werde mir einen Vertrag zuschicken. Diesen Vertrag erhielt ich in dreifacher Ausfertigung. Als ich alle drei Exemplare unterschrieben hatte, schickte ich sie weiter an Mark Pilgrim nach North Carolina (die Deutsche Post berechnete dafür sechs Euro, die es mir aber mehr als wert waren). Parallel dazu erschien auf springer.com bereits „unser" Buch. Der Grafiker (oder die Grafikerin), der/die den Umschlag gestaltet hat, hat wirklich ganze Arbeit geleistet. Sehen Sie doch einfach selbst nach. Die Schlange in den Händen des Männleins ist doch ein toller Verweis auf Python.

Leider war die deutsche (oder die amerikanische) Post nicht in der Lage den Vertrag auszuliefern. Die Gründe dafür werde ich wohl niemals erfahren. Als er auch nach vier Wochen noch nicht bei Mark Pilgrim angekommen war, schickte Frau Glaunsinger den Vertrag per E-Mail an ihn. Kurze Zeit später erhielt ich zwei unterschriebene Exemplare zurück und sendete eines davon unverzüglich nach Heidelberg. Damit war nun also auch rechtlich alles in trockenen Tüchern.

Unterdessen übersetzte ich den restlichen Text und führte schließlich die Formatierung des Manuskripts aus. Die letze Seite stellte ich an einem Samstagmorgen fertig.

Ich kann nur sagen, dass die Arbeit an diesem Buch eine unglaubliche Erfahrung für mich war. Ein halbes Jahr Arbeit meinerseits steckt zwischen den beiden Buchdeckeln. Ein halbes Jahr Nachtschichten, Wochenendarbeit und ab und zu ein klein wenig Ärger mit meinem PC. Ich hoffe, dass sich diese Arbeit gelohnt hat und Sie

einiges aus diesem Buch lernen und in der Praxis anwenden werden. Sie finden in diesem Werk in sehr kompakter Form alle wichtigen Elemente, die Python 3 zu bieten hat.

Über Feedback freue ich mich sehr; besuchen Sie einfach *python3-intensivkurs.de* oder *florianwollenschein.de* und setzen Sie sich mit mir in Verbindung. Wenn Sie Mark Pilgrim etwas zukommen lassen möchten, surfen Sie zu *diveintomark.org*.

Unter *python3-intensivkurs.de* finden Sie außerdem den in diesem Buch verwendeten Beispielcode zum Herunterladen.

Nun wünsche ich Ihnen viel Spaß und Erfolg bei der Arbeit mit diesem Buch.

November 2009 Florian Wollenschein

Danksagung

Danksagung von Mark Pilgrim

Ich danke meiner Frau, die mit ihrer nie enden wollenden Unterstützung und Ermutigung dazu beigetragen hat, dass dieses Buch nicht nur ein weiterer Punkt auf meiner To-Do-Liste ist.

Mein Dank gilt Raymond Hettinger, dessen Alphametik-Löser die Grundlage des achten Kapitels bildet.

Ebenso danke ich Jesse Noller dafür, dass sie mir während der PyCon 2009 so viele Dinge erklärt hat, die ich somit jedem anderen erklären konnte.

Schließlich danke ich den vielen Menschen, die mir während des Schreibens ihr Feedback zukommen ließen, insbesondere gilt mein Dank Giulio Piancastelli, Florian Wollenschein und all den netten Menschen von python.reddit.com.

Danksagung des Übersetzers

Mein Dank gilt zu allererst Mark Pilgrim, der diese Übersetzung mit seinem Buch und seiner unglaublichen Unterstützung während der ganzen Monate überhaupt erst möglich gemacht hat. Thank you very much, Mark.

Außerdem danke ich Hermann Engesser, Dorothea Glaunsinger und Gabriele Fischer vom Springer-Verlag. Alle drei haben das Projekt voller Begeisterung betreut und mir immer und immer wieder Fragen beantwortet.

Es waren harte (und dennoch schöne) Wochen dieses Buch zu übersetzen und so gilt mein Dank auch und ganz besonders Daniela, die mir Tag für Tag unter die Arme gegriffen hat.

Meine Eltern Jürgen und Cornelia, meine Schwester Sarah und meine Oma Margarete darf ich nicht außen vor lassen … Hey, es ist endlich gedruckt!!!

Inhaltsverzeichnis

Kapitel 1
Python installieren

1.1 Los geht's

Willkommen bei Python 3. Lassen Sie uns loslegen. Im Verlauf dieses Kapitels werden Sie die für Sie passende Version von Python 3 installieren.

1.2 Welche Python-Version ist die Richtige für Sie?

Zuerst müssen Sie Python installieren, oder etwa nicht?

Benutzen Sie einen Account auf einem gehosteten Server, so könnte es sein, dass Ihr Provider Python 3 bereits installiert hat. Verwenden Sie zu Hause Linux, dann haben Sie Python 3 vielleicht auch schon. Die allermeisten bekannten GNU/Linux-Distributionen haben Python 2 standardmäßig installiert; eine kleinere, aber steigende Zahl von Distributionen hat zusätzlich auch Python 3 an Bord. (Wie Sie in diesem Kapitel noch erfahren werden, können Sie mehr als eine Python-Version auf Ihrem Computer installiert haben.) Mac OS X besitzt eine Kommandozeilen-Version von Python 2, doch zum Zeitpunkt da ich dies schreibe ist Python 3 nicht enthalten. Microsoft Windows hat von Haus aus gar kein Python dabei. Doch das ist kein Grund zu verzweifeln! Die Installation von Python läuft unter allen Betriebssystemen sehr einfach ab.

Die einfachste Möglichkeit zu überprüfen, ob Python 3 auf Ihrem Linux- oder Mac-OS-X-System bereits installiert ist, besteht darin, eine Kommandozeile zu öffnen. Unter Linux sehen Sie dazu einfach unter Ihrem `Anwendungen`-Menü nach einem Programm namens `Terminal` oder `Konsole`. (Vielleicht finden Sie es auch in einem Untermenü mit dem Namen `Zubehör` oder `System`.) Unter Mac OS X gibt es dazu die Anwendung `Terminal.app` im Ordner `/Programme/Dienstprogramme/`.

Haben Sie die Kommandozeile geöffnet, geben Sie einfach `python3` ein (nur Kleinbuchstaben, keine Leerzeichen) und sehen Sie, was passiert. Auf meinem Linux-System zu Hause ist Python 3 bereits installiert und der Befehl öffnet die interaktive Shell von Python.

M. Pilgrim, *Python 3 – Intensivkurs,*
DOI 10.1007/978-3-642-04377-2_1, © Springer-Verlag Berlin Heidelberg 2010

```
mark@atlantis:~$ python3
Python 3.0.1+ (r301:69556, Apr 15 2009, 17:25:52)
[GCC 4.3.3] on linux2
Type "help", "copyright", "credits" or "license" for more infor-
mation.
>>>
```

(Geben Sie `exit()` ein und drücken Sie `EINGABE`, um die interaktive Shell zu verlassen.)

Mein Webhoster verwendet ebenfalls Linux und bietet Kommandozeilenzugriff; Python 3 ist aber auf meinem Server nicht installiert. (Buh!)

```
mark@manganese:~$ python3
bash: python3: command not found
```

Nun zurück zur Frage, mit der ich diesen Abschnitt begonnen habe: „Welche Python-Version ist die Richtige für Sie?“ Die Antwort: jede, die auf Ihrem Computer läuft.

1.3 Installation unter Microsoft Windows

Windows ist heute in zwei Architekturen erhältlich: 32-Bit und 64-Bit. Natürlich gibt es viele verschiedene Windows-Versionen – XP, Vista, Windows 7 – doch Python läuft unter all diesen Versionen. Es ist wichtiger, zwischen 32-Bit und 64-Bit zu unterscheiden. Wissen Sie nicht, welche Architektur Ihr Windows verwendet, ist es wahrscheinlich 32-Bit.

Laden Sie unter `python.org/download/` die zu Ihrer Architektur passende Version des Windows-Installers herunter. Sie haben dort etwa die folgenden Wahlmöglichkeiten:

- **Python 3.1 Windows installer** (Windows binary – does not include source)
- **Python 3.1 Windows AMD64 installer** (Windows AMD64 binary – does not include source); („does not include source“ bedeutet, dass der Quelltext nicht enthalten ist; Anm. d. Übers.)

Ich verzichte hier darauf, direkte Downloadlinks anzugeben, da ständig kleinere Aktualisierungen hinzukommen und ich nicht dafür verantwortlich sein möchte, dass Sie wichtige Updates verpassen. Sie sollten immer die aktuellste Python 3.x-Version installieren, es sei denn, Sie haben einen Grund dies nicht zu tun.

Ist der Download abgeschlossen, doppelklicken Sie auf die `.msi`-Datei. Windows öffnet daraufhin eine Sicherheitswarnung, da Sie versuchen ausführbaren Code zu starten. Das offizielle Python-Installationsprogramm ist digital signiert von der *Python Software Foundation*, einem Non-Profit-Unternehmen, das die Entwicklung von Python überwacht. Vertrauen Sie nur dem Original!

Klicken Sie auf die `Ausführen`-Schaltfläche, um das Installationsprogramm zu starten.

Zuerst müssen Sie wählen, ob Sie Python 3 für alle Benutzer oder nur für sich installieren möchten. Standardmäßig ist „Für alle Benutzer installieren“ (*Install for all users*) ausgewählt; dies ist auch die beste Wahl, sofern Sie keinen guten Grund haben, eine andere zu treffen. (Ein möglicher Grund, warum Sie „Nur für mich installieren“ (*Install just for me*) wählen würden, ist dass Sie Python auf dem Computer Ihres Unternehmens installieren möchten, aber Ihr Windows-Benutzerkonto keine Administratorrechte besitzt. Doch warum sollten Sie Python ohne die Erlaubnis Ihres Windows-Administrators installieren? Machen Sie mir hier keinen Ärger!)

Klicken Sie auf „Weiter“ (*Next*), um die von Ihnen gewählte Installationsart zu akzeptieren.

Nun fordert das Installationsprogramm Sie auf, ein Zielverzeichnis zu wählen. Das Standardverzeichnis für alle Versionen von Python 3.1.x ist `C:\Python31\`; dieses Verzeichnis sollte nur geändert werden, wenn es nötig ist. Haben Sie einen eigenen Laufwerksbuchstaben auf dem Sie Ihre Anwendungen installieren, können Sie diesen mithilfe der integrierten Schaltflächen suchen oder einfach den Pfadnamen in der unteren Box eingeben. Sie müssen Python nicht auf `C:` installieren, sondern können jedes beliebige Laufwerk und jeden beliebigen Ordner verwenden.

Klicken Sie auf „Weiter“ (*Next*), um Ihre Wahl des Zielverzeichnisses zu akzeptieren.

Das nächste Fenster sieht verwirrend aus, ist es aber eigentlich nicht. Wie bei vielen Installationen können Sie auch hier einzelne Komponenten von Python 3 abwählen. Ist Ihr Festplattenspeicher sehr begrenzt, können Sie bestimmte Teile von der Installation ausschließen.

- **Register Extensions** erlaubt es Ihnen, Python-Skripte (`.py`-Dateien) durch einen Doppelklick zu starten. Empfohlen, aber nicht notwendig. (Diese Option benötigt keinerlei Festplattenspeicher; es macht also wenig Sinn, sie abzuwählen.)
- **Tcl/Tk** ist die Grafikbibliothek die von der Python-Shell, die Sie im Verlauf des Buches benutzen werden, verwendet wird. Ich empfehle dringend, diese Option selektiert zu lassen.
- **Documentation** installiert eine Hilfe-Datei, welche die meisten der Informationen auf `docs.python.org` beinhaltet. Empfohlen, wenn Sie ein Analogmodem verwenden, oder nur begrenzt Zugang zum Internet haben.
- **Utility Scripts** enthält das Skript `2to3.py`, worüber Sie später in diesem Buch mehr erfahren werden. Wird benötigt, wenn Sie lernen möchten, wie man vorhandenen Code von Python 2 zu Python 3 portieren kann. Wenn Sie keinen vorhandenen Python-2-Code haben, können Sie diese Option abwählen.
- **Test Suite** ist eine Skriptsammlung, die verwendet wird, um den Python-Interpreter selbst zu überprüfen. Ich werde sie weder in diesem Buch verwenden, noch habe ich sie jemals während der Programmierung mit Python benutzt. Optional.

Sind Sie unsicher, wie viel freien Festplattenspeicher Sie haben, können Sie auf die Schaltfläche „Disk Usage“ klicken. Das Installationsprogramm listet daraufhin

alle Laufwerke auf, berechnet den verfügbaren Speicherplatz jedes Laufwerks und zeigt, wie viel Platz nach der Installation übrig bleiben würde.

Klicken Sie auf OK, um zum Fenster „Customize Python“ zurückzukehren.

Entscheiden Sie sich, eine Option abzuwählen, klicken Sie auf die Schaltfläche mit einer abgebildeten Festplatte und wählen Sie „Entire feature will be unavailable“. Wählen Sie z. B. die Test Suite ab, so sparen Sie gigantische 7.908 KB Speicherplatz.

Klicken Sie auf „Weiter“ (*Next*), um Ihre Auswahl zu akzeptieren.

Das Installationsprogramm kopiert nun alle notwendigen Dateien in das von Ihnen gewählte Zielverzeichnis.

Klicken Sie auf „Beenden“ (*Finish*), um die Installation abzuschließen und das Installationsprogramm zu schließen.

In Ihrem Startmenü sollte sich jetzt ein Eintrag mit der Bezeichnung `Python 3.1` finden. Gehen Sie mit dem Mauszeiger darüber, finden Sie darin ein Programm mit dem Namen `IDLE`. Wählen Sie dieses Programm aus, um die interaktive Python-Shell zu starten.

1.4 Installation unter Mac OS X

Jeder moderne Macintosh-Computer verwendet den Intel-Chip (wie die meisten Windows PCs). Ältere Macs benutzten dagegen PowerPC-Chips. Sie müssen den Unterschied zwischen diesen Chips nicht kennen, da es nur ein Mac-Python-Installationsprogramm für alle Macs gibt.

Rufen Sie `python.org/download/` auf und laden Sie das Mac-Installationsprogramm herunter. Die Bezeichnung lautet etwa **Python 3.1 Mac Installer Disk Image**, wobei die Versionsnummer abweichen kann. Stellen Sie jedoch sicher, dass Sie Version 3.x herunterladen und nicht 2.x.

Ihr Browser sollte das Speicherabbild automatisch einhängen und den Inhalt in einem Finder-Fenster anzeigen. (Geschieht dies nicht, müssen Sie das Speicherabbild in Ihrem Download-Ordner suchen und zum Einhängen darauf doppelklicken. Die Bezeichnung lautet etwa `python-3.1.dmg`.) Das Speicherabbild enthält einige Textdateien (`Build.txt`, `License.txt`, `ReadMe.txt`) und das eigentliche Installationspaket `Python.mpkg`.

Doppelklicken Sie auf das `Python.mpkg`-Installationspaket, um das Mac-Python-Installationsprogramm zu starten.

Das erste Fenster enthält eine kurze Beschreibung von Python selbst und verweist Sie auf die `ReadMe.txt`-Datei (die Sie nicht gelesen haben, richtig?), um weitere Details zu erfahren.

Klicken Sie auf „Fortfahren“ (*Continue*).

Das nächste Fenster enthält nun einige wichtige Informationen: Python setzt Mac OS X 10.3 oder höher voraus. Verwenden Sie immer noch Mac OS X 10.2, dann sollten Sie wirklich auf eine neuere Version umsteigen. Apple stellt für Ihr Betriebssystem nicht länger Sicherheitsaktualisierungen bereit, und Ihr Computer

könnte gefährdet sein, wenn Sie online gehen. Außerdem können Sie Python 3 nicht verwenden.

Klicken Sie auf „Fortfahren“ (*Continue*), um zum nächsten Schritt zu gelangen.

Pythons Installationsprogramm zeigt – wie alle guten Installationsprogramme – den Softwarelizenzvertrag an. Python ist Open Source und seine Lizenz von der *Open Source Initiative* anerkannt. Python hatte in seiner Geschichte einige Eigentümer und Förderer, von denen jeder seine Spuren in der Softwarelizenz hinterlassen hat. Doch das Endresultat ist dieses: Python ist Open Source und Sie können es auf jeder beliebigen Plattform, zu jedem beliebigen Zweck, ohne Gebühr oder Verpflichtungen nutzen.

Klicken Sie abermals auf „Fortfahren“ (*Continue*).

Aufgrund einer Eigenart des Apple Installations-Frameworks müssen Sie der Softwarelizenz „zustimmen”, wenn Sie die Installation abschließen möchten. Da Python Open Source ist, „stimmen Sie zu“, dass die Lizenz Ihnen zusätzliche Rechte gewährt, statt sie Ihnen wegzunehmen.

Klicken Sie zum Fortfahren auf „Annehmen“ (*Agree*).

Das nächste Fenster erlaubt es Ihnen, den Installationsort zu verändern. Sie müssen Python auf Ihrem Bootlaufwerk installieren, was das Installationsprogramm aufgrund von Begrenzungen aber nicht forciert.

Aus diesem Fenster heraus können Sie außerdem Ihre Installation anpassen und bestimmte Funktionen ausschließen. Klicken Sie auf *Customize*, wenn Sie dies tun möchten. Andernfalls klicken Sie auf *Install*.

Haben Sie eine angepasste Installation gewählt, stellt Ihnen das Installationsprogramm die folgende Liste von Funktionen zur Verfügung:

- **Python Framework** Dies ist das Herzstück von Python und sowohl ausgewählt, als auch deaktiviert, da es installiert werden muss.
- **GUI Applications** beinhaltet `IDLE`, die grafische Python-Shell, die Sie im Verlauf dieses Buches verwenden werden. Ich empfehle dringend, diese Option ausgewählt zu lassen.
- **UNIX command-line tools** beinhaltet die Kommandozeilenanwendung `python3`. Auch hier empfehle ich dringend, die Option ausgewählt zu lassen.
- **Python Documentation** beinhaltet die meisten der Informationen von `docs.python.org`. Empfohlen, wenn Sie ein Analogmodem verwenden, oder nur begrenzt Zugang zum Internet haben.
- **Shell profile updater** überwacht, ob Ihr Shell-Profil (das in `Terminal.app` Verwendung findet) aktualisiert werden muss, um sicherzustellen, dass diese Version von Python sich im Suchpfad Ihrer Shell befindet. Sie müssen dies wahrscheinlich nicht ändern.
- **Fix system Python** sollte nicht geändert werden. (Es teilt Ihrem Mac mit, dass er Python 3 als Standard für alle Python-Skripts verwenden soll, auch für die von Apple ins System integrierte. Dies wäre sehr schlecht, da die meisten dieser Skripts in Python 2 geschrieben sind und unter Python 3 nicht richtig laufen würden.)

Klicken Sie auf *Install*, um fortzufahren.

Da das Installationsprogramm systemweite Frameworks und ausführbare Programme in `/usr/local/bin/` installiert, fragt es nach Ihrem Administratorpasswort. Es besteht keine Möglichkeit, Mac Python ohne Administratorrechte zu installieren.

Klicken Sie auf OK, um die Installation zu starten.

Es wird ein Fortschrittsbalken angezeigt, während die von Ihnen gewählten Funktionen installiert werden.

Ist alles gut gelaufen, wird Ihnen ein großes, grünes Häkchen angezeigt, das Ihnen die erfolgreiche Installation bestätigt.

Klicken Sie auf *Close*, um das Installationsprogramm zu beenden.

Haben Sie den Installationsort nicht verändert, so finden Sie die gerade installierten Dateien im Ordner `Python 3.1` Ihres `/Anwendungen`-Ordners. Am wichtigsten ist hier `IDLE`, die grafische Python-Shell.

Doppelklicken Sie auf `IDLE`, um die Python-Shell zu starten.

Die Python-Shell ist das Programm, mit dem Sie während des Erkundens von Python die meiste Zeit verbringen werden. Die Beispiele in diesem Buch gehen davon aus, dass Sie wissen, wie Sie die Python-Shell starten.

1.5 Installation unter Ubuntu Linux

Moderne Linux-Distributionen werden durch sogenannte Repositorys mit einem enormen Umfang an vorkompilierten Anwendungen unterstützt, die einfach installiert werden können. Unter Ubuntu Linux besteht die einfachste Möglichkeit zur Installation von Python 3 darin, die Anwendung `Hinzufügen/Entfernen` in Ihrem `Anwendungen`-Menü zu verwenden.

Nach dem Start der `Hinzufügen/Entfernen`-Anwendung wird Ihnen eine Liste vorselektierter Anwendungen in unterschiedlichen Kategorien angezeigt. Einige sind schon installiert; die meisten jedoch nicht. Da das Repository über 10.000 Anwendungen enthält, existieren verschiedene Filter, die Sie anwenden können, um nur kleine Teile des Repositorys zu sehen. Als Standardfilter ist „Canonical-maintained applications“ ausgewählt, welcher eine kleine Untermenge aller Anwendungen die offiziell von Canonical – dem Unternehmen, das Ubuntu Linux entwickelt und pflegt – unterstützt werden, anzeigt.

Da Python 3 nicht von Canonical gepflegt wird, müssen Sie zuerst einmal das Filtermenü aufklappen und „All Open Source applications“ auswählen.

Haben Sie den Filter so erweitert, dass er alle Open-Source-Anwendungen anzeigt, benutzen Sie die Suche direkt neben dem Filtermenü, um nach Python 3 zu suchen.

Die Liste beschränkt sich nun nur noch auf die Anwendungen, die Python 3 beinhalten. Sie werden zwei Pakete auswählen. Das erste Paket ist `Python (v3.0)`. Dieses enthält den Python-Interpreter selbst.

Das zweite gewünschte Paket finden Sie direkt darüber: `IDLE (using Python-3.0)`. Dies ist eine grafische Shell für Python, die Sie im Verlauf dieses Buches nutzen werden.

Haben Sie diese beiden Pakete ausgewählt, klicken Sie auf *Änderungen anwenden*, um fortzufahren.

Der Paketmanager bittet Sie nun, zu bestätigen, dass Sie `IDLE (using Python-3.0)` und `Python (v3.0)` installieren möchten.

Klicken Sie zum Fortfahren auf *Anwenden*.

Der Paketmanager zeigt Ihnen einen Fortschrittsbalken an, während er die notwendigen Pakete von Canonicals Internet-Repository herunterlädt.

Sind die Pakete fertig heruntergeladen, beginnt der Paketmanager automatisch mit der Installation.

Ist alles korrekt verlaufen, bestätigt Ihnen der Paketmanager, dass beide Pakete erfolgreich installiert wurden. Hier können Sie mit einem Doppelklick auf `IDLE` die Python-Shell starten, oder den Paketmanager verlassen, indem Sie auf *Schließen* klicken.

Sie können die Python-Shell immer starten, indem Sie in Ihrem `Anwendungen`-Menü im Untermenü `Entwicklung` `IDLE` auswählen.

In der Python-Shell werden Sie beim Erkunden von Python die meiste Zeit verbringen. Die in diesem Buch aufgeführten Beispiele setzen voraus, dass Sie wissen, wie man die Python-Shell startet.

1.6 Installation auf anderen Plattformen

Python 3 ist für einige verschiedene Plattformen verfügbar. Im Besonderen für nahezu jede Linux-, BSD- und Solaris-basierte Distribution. RedHat Linux verwendet beispielsweise den Paketmanager `yum`; FreeBSD hat seine Ports und Paketsammlungen; Solaris benutzt `pkgadd` und Konsorten. Eine kurze Websuche nach `Python 3 + Ihr Betriebssystem` wird Ihnen zeigen, ob Python 3 dafür verfügbar ist und wenn ja, wie man es installiert.

1.7 Verwenden der Python-Shell

Die Python-Shell ist das Programm, in dem Sie die Syntax von Python erkunden, interaktive Hilfe zu Befehlen erhalten und kurze Programme debuggen können. Die grafische Python-Shell (`IDLE` genannt) verfügt außerdem über einen Texteditor, der die farbige Hervorhebung der Python-Syntax unterstützt und sich in die Python-Shell integriert. Haben Sie noch keinen bevorzugten Texteditor, sollten Sie `IDLE` ausprobieren.

Eins nach dem anderen. Die Python-Shell selbst ist eine unglaubliche interaktive Spielwiese. Im Verlauf dieses Buches werden Ihnen Beispiele wie dieses begegnen:

```
>>> 1 + 1
2
```

Die drei spitzen Klammern, >>>, stellen die Eingabeaufforderung der Python-Shell dar. Geben Sie diese nicht ein. Sie dienen nur dazu, Ihnen zu zeigen, dass Sie nach der Eingabeaufforderung etwas eingeben sollen.

`1 + 1` ist der Teil, den Sie eingeben müssen. Sie können jeden beliebigen gültigen Python-Ausdruck oder -Befehl in der Python-Shell eingeben. Scheuen Sie sich nicht; sie wird nicht beißen! Das Schlimmste was passieren kann, ist dass Sie eine Fehlermeldung erhalten. Befehle werden sofort ausgeführt (sobald Sie `EINGABE` drücken); Ausdrücke werden sofort ausgewertet und die Python-Shell zeigt das Ergebnis an.

`2` ist das Ergebnis des ausgewerteten Ausdrucks. Zufällig ist `1 + 1` ein gültiger Python-Ausdruck. Das Ergebnis ist natürlich `2`.

Lassen Sie uns noch etwas versuchen.

```
>>> print('Hello world!')
Hello world!
```

Ziemlich einfach, nicht wahr? Doch Sie können in der Python-Shell sehr viel mehr machen. Sollten Sie jemals stecken bleiben – Sie können sich an einen Befehl oder an die passenden Argumente einer bestimmten Funktion nicht mehr erinnern – können Sie innerhalb der Python-Shell interaktive Hilfe erhalten. Geben Sie dazu einfach `help` ein und drücken Sie `EINGABE`.

```
>>> help
Type help() for interactive help, or help(object) for help about
object.
```

Die Python-Shell bietet zwei Formen der Hilfe an. Die Hilfe zu einem einzelnen Objekt gibt einfach die Dokumentation aus und führt Sie zurück zur Eingabeaufforderung. Sie können aber auch in den Hilfemodus wechseln, was dazu führt, dass nun keine Python-Ausdrücke mehr ausgewertet werden, sondern Ihnen bei Eingabe eines Keywords oder Befehls alles angezeigt wird, was die Hilfe darüber weiß.

Um in den interaktiven Hilfemodus zu gelangen, geben Sie `help()` ein und drücken Sie `EINGABE`.

```
>>> help()
Welcome to Python 3.0!  This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics".  Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help>
```

Beachten Sie, dass sich die Eingabeaufforderung von >>> in help> ändert. Dies erinnert Sie daran, dass Sie sich im interaktiven Hilfemodus befinden. Nun können Sie jedes beliebige Keyword, jeden beliebigen Befehl, Modulnamen, Funktionsnamen – so ziemlich alles, was Python kennt – eingeben und die Dokumentation dazu lesen.

```
help> print                                                    ①
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a  file-like  object  (stream);  defaults  to  the  current
sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.

help> PapayaWhip                                               ②
no Python documentation found for 'PapayaWhip'

help> quit                                                     ③

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
>>>                                                            ④
```

① Um die Dokumentation zur `print()`-Funktion zu erhalten, geben Sie `print` ein und drücken Sie `EINGABE`. Der interaktive Hilfemodus zeigt etwas an, das einer *man page* ähnelt: den Funktionsnamen, eine kurze Zusammenfassung, die Funktionsargumente und ihre Standardwerte usw. Sollte Ihnen die Dokumentation schleierhaft erscheinen, machen Sie sich bitte keine Sorgen. Im Verlauf der nächsten Kapitel werden Sie mehr darüber erfahren.

② Der interaktive Hilfemodus kennt natürlich nicht alles. Geben Sie etwas ein, das weder ein Python-Befehl, noch ein Python-Modul, noch eine Python-Funktion, oder ein integriertes Keyword ist, zuckt der interaktive Hilfemodus nur mit seinen virtuellen Achseln.

③ Um den interaktiven Hilfemodus zu beenden, geben Sie `quit` ein und drücken Sie `EINGABE`.

④ Die Eingabeaufforderung ändert sich wieder zu >>>, um Ihnen anzuzeigen, dass Sie den interaktiven Hilfemodus verlassen haben und sich wieder in der Python-Shell befinden.

`IDLE`, die grafische Python-Shell, enthält auch einen Python-sensitiven Texteditor. Im nächsten Kapitel erfahren Sie, wie Sie diesen verwenden.

1.8 Python-Editoren und -IDEs

`IDLE` ist nicht das Maß aller Dinge, wenn es darum geht, Programme in Python zu schreiben. `IDLE` ist hilfreich zum Erlernen der Sprache selbst, doch viele Entwickler bevorzugen andere Texteditoren oder Integrierte Entwicklungsumgebungen (engl. *Integrated Development Environment*, kurz *IDE*; Anm. d. Übers.). Ich werde diese hier nicht behandeln, doch die Python-Community pflegt eine Liste von Python-sensitiven Editoren, die eine große Bandbreite an unterstützten Plattformen und Softwarelizenzen abdecken.

Eine IDE die Python 3 unterstützt ist PyDev, ein Plug-in für Eclipse, das Eclipse in eine ausgewachsene Python-IDE verwandelt. Sowohl Eclipse, als auch PyDev sind plattformübergreifend und Open Source.

Auf kommerzieller Seite hat Komodo IDE von ActiveState die Nase vorn. Man muss zwar pro Benutzer eine Lizenz kaufen, aber es gibt einen Studentenrabatt und eine kostenlose zeitlimitierte Demoversion.

Ich programmiere nun seit neun Jahren mit Python und ich bearbeite meine Programme mit GNU Emacs und debugge sie in der Python-Shell auf der Kommandozeile. Es gibt kein richtig oder falsch bei der Auswahl der Werkzeuge. Finden Sie Ihren eigenen Weg!

Kapitel 2
Ihr erstes Python-Programm

2.1 Los geht's

Bücher über das Programmieren beginnen meist mit einer Reihe langweiliger Kapitel über die Grundlagen, bis schlussendlich ein sinnvolles Programm herauskommt. Lassen Sie uns all das überspringen. Nachfolgend finden Sie ein komplettes, funktionierendes Python-Programm. Sie werden es vielleicht nicht sofort verstehen, doch machen Sie sich darüber keine Sorgen, denn wir werden es Zeile für Zeile auseinandernehmen. Sehen Sie es sich erst einmal an. Vielleicht verstehen Sie ja doch etwas.

```
SUFFIXES = {1000: ['KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'],
            1024: ['KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB']}

def approximate_size(size, a_kilobyte_is_1024_bytes=True):
    '''Convert a file size to human-readable form.

    Keyword arguments:
    size -- file size in bytes
    a_kilobyte_is_1024_bytes -- if True (default), use multiples of 1024
                                if False, use multiples of 1000

    Returns: string

    '''
    if size < 0:
        raise ValueError('number must be non-negative')

    multiple = 1024 if a_kilobyte_is_1024_bytes else 1000
    for suffix in SUFFIXES[multiple]:
        size /= multiple
        if size < multiple:
            return '{0:.1f} {1}'.format(size, suffix)
```

M. Pilgrim, *Python 3 – Intensivkurs,*
DOI 10.1007/978-3-642-04377-2_2, © Springer-Verlag Berlin Heidelberg 2010

```
    raise ValueError('number too large')


if __name__ == '__main__':
    print(approximate_size(1000000000000, False))
    print(approximate_size(1000000000000))
```

Wenn wir dieses Programm nun über die Kommandozeile starten, sieht das unter Windows etwa so aus:

```
c:\home\diveintopython3> c:\python30\python.exe humansize.py
1.0 TB
931.3 GiB
```

Unter Mac OS X oder Linux sollte es wie folgt aussehen:

```
you@localhost:~$ python3 humansize.py
1.0 TB
931.3 GiB
```

Was ist da gerade passiert? Sie haben Ihr erstes Python-Programm ausgeführt. Sie haben den Python-Interpreter auf der Kommandozeile aufgerufen und ihm den Namen des Skripts übergeben, das Sie ausführen wollten. Dieses Skript definiert die Funktion `approximate_size()`, die eine genaue Dateigröße in Bytes übernimmt und eine „schönere“ (aber ungenauere) Größe berechnet.

Am Ende dieses Skripts sehen Sie zwei aufeinanderfolgende Aufrufe von `print(approximate_size(arguments))`. Dies sind Funktionsaufrufe; zunächst wird die `approximate_size()`-Funktion mit einigen übergebenen Argumenten aufgerufen, dann wird der Rückgabewert an die `print()`-Funktion übergeben. Die `print()`-Funktion ist eine integrierte Funktion. Sie können sie jederzeit, überall verwenden. (Es gibt sehr viele integrierte Funktionen, aber auch sehr viele Funktionen, die sich in Modulen befinden. Geduld, mein Freund.)

Warum gibt das Skript auf der Kommandozeile immer dasselbe aus? Dazu kommen wir noch. Erst einmal sehen wir uns die `approximate_size()`-Funktion an.

2.2 Funktionen deklarieren

Python nutzt, ebenso wie die meisten anderen Programmiersprachen, Funktionen. Es existieren jedoch keine separaten Header-Dateien wie in C++, oder `interface`/`implementation`-Abschnitte wie in Pascal. Wenn Sie eine Funktion benötigen, deklarieren Sie sie einfach wie folgt:

```
def approximate_size(size, a_kilobyte_is_1024_bytes=True):
```

Mit `def` wird die Funktionsdeklaration begonnen; darauf folgt der Name der Funktion und die Argumente eingeschlossen in Klammern. Mehrere Argumente werden durch Kommas getrennt.

Beachten Sie auch, dass die Funktion keinen Datentyp für einen Rückgabewert definiert. In Python geben Funktionen den Datentyp ihres Rückgabewertes nicht an; es wird nicht einmal angegeben, ob sie überhaupt einen Wert zurückgeben. Tatsächlich gibt jedoch jede Funktion in Python einen Wert zurück; sollte die Funktion eine `return`-Anweisung enthalten, wird der dort angegebene Wert zurückgegeben, ansonsten liefert sie `None` zurück, den Nullwert von Python.

☞ In manchen Sprachen beginnen Funktionen (die einen Wert zurückgeben) mit `function`; Subroutinen (die keinen Wert zurückgeben) beginnen mit `sub`. In Python gibt es keine Subroutinen. Alles ist eine Funktion, alle Funktionen geben einen Wert zurück (auch wenn es `None` ist) und alle Funktionen beginnen mit `def`.

Die Funktion `approximate_size` nimmt zwei Argumente entgegen – `size` und `a_kilobyte_is_1024_bytes` – doch keines der Argumente definiert einen Datentyp. (Wie Sie vielleicht an der `=True`-Syntax erkennen, handelt es sich bei dem zweiten Argument um einen booleschen Wert. Datentypen werden in Python niemals explizit angegeben. Python findet automatisch heraus, welchen Typ eine Variable hat und überwacht dies intern.)

☞ In Java, C++ und anderen statisch-typisierten Sprachen müssen Sie einen Datentyp für die Funktion, den Rückgabewert und jedes Argument angeben. In Python werden niemals explizit Datentypen für irgendetwas angegeben. Python überwacht den Datentyp intern, basierend auf dem Wert, den Sie zuweisen.

2.2.1 Pythons Datentypen im Vergleich mit denen anderer Sprachen

Ein pfiffiger Programmierer hat mir folgenden Vergleich von Python mit anderen Sprachen zugesendet.

Statisch typisierte Sprache Eine Programmiersprache, bei der Datentypen zur Kompilierzeit feststehen. Die Mehrheit der statisch typisierten Sprachen sorgen dafür, indem Sie den Programmierer zwingen, allen Variablen vor der Verwendung einen Datentyp zuzuweisen. Java und C sind statisch typisierte Programmiersprachen.

Dynamisch typisierte Sprache Eine Programmiersprache, bei der der Datentyp zur Laufzeit ermittelt wird; das Gegenteil von statisch typisiert. VBScript und Python sind dynamisch typisierte Sprachen, die den Datentyp einer Variablen bei der ersten Zuweisung ermitteln.

Stark typisierte Sprache Eine Programmiersprache, bei der Datentypen immer erzwungen werden. Java und Python sind stark typisiert. Sie können zum Beispiel eine Ganzzahl ohne explizite Konvertierung nicht wie einen String verwenden.

Schwach typisierte Sprache Eine Programmiersprache, bei der Datentypen ignoriert werden können; das Gegenteil von stark typisiert. VBScript ist schwach typisiert, da man zum Beispiel den String `'12'` und die Ganzzahl `3` zum String `'123'`